Amendments to the Claims

The Claim Listing below replaces all prior versions of the claims in the subject application.

Claim Listing:

Claims 1 - 60 (cancelled).

Please add the following new claims 61 to 76:

61 (new). Computer-readable storage medium storing instructions that when executed result in a processor performing operations comprising:

assigning received packets to threads for processing, the threads to be executed by microengines in the processor, each respective microengine to employ respective local event signals and global signaling states, the respective local event signals being local to respective threads executed by the respective microengine and being to provide information on whether functions requested by the respective threads have completed, the global signaling states permitting an executing thread signal state to be broadcast to the microengines, the threads executed by the microengines being able to branch based on the global signaling states, the threads executed by the microengines maintain thread capability information and port-to-thread assignments; and

in event of a processing exception concerning a particular received packet, sending the particular received packet to a processor core for further processing by the processor core, functionality of the threads executed by the microengines being determined by microcode loaded via the processor core into the microengines.

62 (new). The computer-readable storage medium of claim 61, wherein:

all of the threads executed by the microengines are able to branch based on the global signaling states; and

the global signaling states are used to determine one of:

resource availability; and

whether resource servicing is due.

63 (new). The computer-readable storage medium of claim 61, wherein:

the threads executed by the microengines include a scheduler thread to maintain the thread capability information and the port-to-thread assignments; and

the scheduler thread is also to maintain thread busy tracking information that indicates certain threads that are actively servicing a port.

64 (new). The computer-readable storage medium of claim 63, wherein:

the thread capability information indicates to the scheduler thread capabilities of other threads and which of the other threads may be appropriate to service a particular port; and

the scheduler thread uses the thread capability information, the port-to-thread assignments, and the thread busy tracking information to perform the assigning of the received packets.

65 (new). The computer-readable storage medium of claim 61, wherein:

the threads executed by the microengines include a processing thread to parse a header of the particular received packet and to perform a lookup based on the header;

unless the processing exception occurs, the processing thread is to store the particular received packet in memory and to enqueue the particular received packet by placing a packet link descriptor for the particular received packet in a transmit queue associated with a forwarding port indicated by the lookup; and

if the processing exception occurs, the processing thread is to perform the sending of the particular received packet.

66 (new). The computer-readable storage medium of claim 65, wherein:

the threads executed by the microengines also include a transmit processing thread, an arbiter thread, and a transmit scheduler thread;

the transmit scheduler thread is to assign the particular received packet to the transmit processing thread;

the transmit processing thread is to send the particular received packet to the forwarding port; and

the arbiter thread is to prioritize transmission queues.

67 (new). The computer-readable memory of claim 61, wherein:

the processor core and the microengines are coupled via an input/output (I/O) bus interface to ports to receive the received packets; and

the processor core to make function calls through an operating system to operate on the microengines.

68 (new). The computer-readable memory of claim 61, wherein the respective microengine includes:

general purpose registers coupled to an arithmetic logic unit;

multiple program counters coupled to a control store to receive the microcode loaded into the respective microengine via the processor core; and

context switching logic coupled to the program counters.

69 (new). A method comprising:

assigning received packets to threads for processing, the threads to be executed by microengines in a processor, each respective microengine to employ respective local event signals and global signaling states, the respective local event signals being local to respective threads executed by the respective microengine and being to provide information on whether functions requested by the respective threads have completed, the global signaling states permitting an executing thread signal state to be broadcast to the microengines, the threads executed by the microengines being able to branch based on the

global signaling states, the threads executed by the microengines maintain thread capability information and port-to-thread assignments; and

in event of a processing exception concerning a particular received packet, sending the particular received packet to a processor core for further processing by the processor core, functionality of the threads executed by the microengines being determined by microcode loaded via the processor core into the microengines.

70 (new). The method of claim 69, wherein:

all of the threads executed by the microengines are able to branch based on the global signaling states; and

the global signaling states are used to determine one of:

resource availability; and

whether resource servicing is due.

71 (new). The method of claim 69, wherein:

the threads executed by the microengines include a scheduler thread to maintain the thread capability information and the port-to-thread assignments; and

the scheduler thread is also to maintain thread busy tracking information that indicates certain threads that are actively servicing a port.

72 (new). The method of claim 71, wherein:

the thread capability information indicates to the scheduler thread capabilities of other threads and which of the other threads may be appropriate to service a particular port; and

the scheduler thread uses the thread capability information, the port-to-thread assignments, and the thread busy tracking information to perform the assigning of the received packets.

73 (new). The method of claim 69, wherein:

the threads executed by the microengines include a processing thread to parse a header of the particular received packet and to perform a lookup based on the header;

unless the processing exception occurs, the processing thread is to store the particular received packet in memory and to enqueue the particular received packet by placing a packet link descriptor for the particular received packet in a transmit queue associated with a forwarding port indicated by the lookup; and

if the processing exception occurs, the processing thread is to perform the sending of the particular received packet.

74 (new). The method of claim 73, wherein:

the threads executed by the microengines also include a transmit processing thread, an arbiter thread, and a transmit scheduler thread;

the transmit scheduler thread is to assign the particular received packet to the transmit processing thread;

the transmit processing thread is to send the particular received packet to the forwarding port; and

the arbiter thread is to prioritize transmission queues.

75 (new). The method of claim 69, wherein:

the processor core and the microengines are coupled via an input/output (I/O) bus interface to ports to receive the received packets; and

the processor core to make function calls through an operating system to operate on the microengines.

76 (new). The method of claim 69, wherein the respective microengine includes:

general purpose registers coupled to an arithmetic logic unit;

multiple program counters coupled to a control store to receive the microcode loaded into the respective microengine via the processor core; and

context switching logic coupled to the program counters.